```
(*$S+*)
PROGRAM MONITOR;

(*##################################################################*)
(*ATARI Pascal EXECUTION MONITOR                                    *)
(*Version 1.0 :01-Mar-82                                            *)
(*The final delivered version!!! Hooray                             *)
(*Written by: W. Saville and M. Lehman                              *)
(*                                                                  *)
(*##################################################################*)



TYPE
  FILENAME = STRING[16];

   (* global area used by all phases of the compiler and the monitor *)

 (* ATARI GLOBAL RECORD *)
 GBL = RECORD
         FLAG : BOOLEAN;(* IF TRUE COMPILER CALLED FROM MONITOR *)
         PASTITLE : STRING[16];(* INPUT FILE TITLE *)
         TOKTITLE : STRING[16];(* TOKEN FILE TITLE *)
         OUTTITLE : STRING[16](* OUTPUT FILE TITLE *)
       END;

 PAOC3 = PACKED ARRAY [1..3] OF CHAR;

VAR
  COMPTITLE,
  TEMPSTRING,
  EDITTITLE,
  LINKTITLE:STRING[16];

  GLOBALS : ABSOLUTE[$1F80] GBL;
  CMDSTRG : STRING[2];
  CHAINFILE : FILE;
  FNAME : FILENAME;
  IOR : INTEGER;
  TSTEXT,EXT : PAOC3;
  PROMPT : PACKED ARRAY [1..6] OF CHAR;
  NOSCAN : BOOLEAN;


FUNCTION GETNAME(VAR F,PREV:FILENAME; EXT:PAOC3):BOOLEAN;
(* READ IN A FILENAME WITH POSSIBLE DEFAULTS *)

BEGIN
  GETNAME := FALSE;

  REPEAT
    WRITE(PROMPT,' file name? ');
    IF LENGTH(PREV) <> 0 THEN
      BEGIN
WRITELN;
Write('ENTER <return> for default: ')
      END;

    READLN(F);
  UNTIL (LENGTH(F)<> 0) OR
        ((LENGTH(F)= 0) AND (LENGTH(PREV) <> 0));
  IF LENGTH(F)=1 THEN
```

```
      EXIT;

  IF LENGTH(F) = 0 THEN
    BEGIN
      MOVE(PREV,F,LENGTH(PREV)+1);
      MOVE(EXT,F[LENGTH(F)-2],3);
    END
  ELSE (* SEE IF IT NEEDS D: *)
    IF (LENGTH(F) < 3) OR ((F[2] <> ':') AND (F[3] <> ':')) THEN
      F := CONCAT('D:',F);
  GETNAME := TRUE;
END;


FUNCTION COMPNAMES(TOTAL:BOOLEAN):BOOLEAN;

(* READ IN FILE NAMES FOR COMPILATION TOTAL=TRUE IF NOT SCANNING ONLY *)

BEGIN
  COMPNAMES := FALSE;

  PROMPT := 'Source';
  EXT := '   ';
  FNAME := '';
  IF NOT GETNAME(GLOBALS.PASTITLE,FNAME,EXT) THEN
    EXIT;
  IF GLOBALS.PASTITLE[LENGTH(GLOBALS.PASTITLE)-3] <> '.' THEN
    GLOBALS.PASTITLE := CONCAT(GLOBALS.PASTITLE,'.PAS');

  MOVE(GLOBALS.PASTITLE[ORD(GLOBALS.PASTITLE[0])-2],TSTEXT,3);
  EXT := 'TOK';

  NOSCAN := FALSE;
  PROMPT := 'Token ';
  EXT := 'TOK';
  FNAME := GLOBALS.PASTITLE;
  IF NOT NOSCAN THEN
    IF NOT GETNAME(GLOBALS.TOKTITLE,FNAME,EXT) THEN
      EXIT;

  IF TOTAL THEN
    BEGIN
      PROMPT := 'Code  ';
      EXT := 'ERL';
      FNAME := GLOBALS.TOKTITLE;
      IF NOT GETNAME(GLOBALS.OUTTITLE,FNAME,EXT) THEN
EXIT
    END;
  COMPNAMES := TRUE;

END;


PROCEDURE MENU;
BEGIN

WRITE('}');
  WRITELN('       ATARI Pascal      ');
  WRITELN('  VERSION 1.0 : 1-MAR-82');
  WRITELN('     (c) 1982 by ATARI    ');
  WRITELN;
  WRITELN;
  WRITELN('  L)ink          R)un');
  WRITELN('  D)os           Q)uit');
  WRITELN;
  WRITE  ('  Enter letter and <return>:');
END;


PROCEDURE INITNAMES;
BEGIN
        COMPTITLE := 'D:PHO';
        EDITTITLE := 'D2:MEDIT';
        LINKTITLE := 'D:LINK';
END;


PROCEDURE GETCOMMAND;
BEGIN
    REPEAT(* until valid command *)
      REPEAT(* until non-null input *)         MENU;
        READLN(CMDSTRG)
      UNTIL LENGTH(CMDSTRG)=1;
    UNTIL CMDSTRG[1] IN ['C','c','E','e','R','r','L','l','D','d','Q','q'];
END;

BEGIN
   INITNAMES;
  REPEAT
    GETCOMMAND;
  GLOBALS.FLAG := FALSE;

  CASE CMDSTRG[1] OF
        'q','Q',
        'd','D' : BEGIN
                    INLINE($D9);  (* p-code halt instruction; d.b.g. *)
                  END;

  'c','C' : BEGIN
      GLOBALS.FLAG := TRUE;
      IF COMPNAMES(TRUE) THEN
        BEGIN
        WRITELN('  Change D1 to compiler disk');
        WRITELN('  Then type <return>');
        READLN;
        OPEN(CHAINFILE,COMPTITLE,IOR);
        WRITELN('  Loading Compiler...');
          IF IOR = 0 THEN
    CHAIN(CHAINFILE)
  ELSE
        BEGIN
    WRITELN('  ',COMPTITLE,' not found');
    WRITELN('  Change back to Pascal disk');
    WRITELN('  Then type <return>');
    READLN
        END;
  CLOSE(CHAINFILE,IOR)(* TO FREE UP IOCB *)


END;


  'r','R' : BEGIN
      WRITELN('  Enter program name');
      READLN(FNAME);
      IF (LENGTH(FNAME) < 3) OR ((FNAME[2] <> ':') AND (FNAME[3] <> ':')) THEN
        FNAME := CONCAT('D:',FNAME);
      OPEN(CHAINFILE,FNAME,IOR);
      IF IOR = 0 THEN
        CHAIN(CHAINFILE)
      ELSE
        BEGIN
          CLOSE(CHAINFILE,IOR);
          OPEN(CHAINFILE,CONCAT(FNAME,'.COM'),IOR);
          IF IOR = 0 THEN
                CHAIN(CHAINFILE)
             ELSE
                BEGIN
                WRITELN('  ',FNAME,' not found');
                WRITELN('  Check your program name');
                WRITELN('  Then type <return>');
                READLN;
                CLOSE(CHAINFILE,IOR)  (* TO FREE UP IOCB *)
                END
        END
END;

   'e','E' : BEGIN
      WRITELN('  Loading Editor...');
      OPEN(CHAINFILE,EDITTITLE,IOR);
      IF IOR = 0 THEN
        CHAIN(CHAINFILE)
      ELSE
WRITELN('  ',EDITTITLE,' not found');
      CLOSE(CHAINFILE,IOR);(* TO FREE UP IOCB *)
      WRITELN('  Type <return>');
      READLN;
    END;

   '1','L' : BEGIN
        WRITELN('  Loading Linker');
        WRITELN('  When Linker prompts with "*" enter');
        WRITELN('  your .ERL file names separated by');
        WRITELN('  commas ending with PASLIB/S');
        WRITELN;
        WRITELN('  Then type <return>');
        WRITELN;
        OPEN(CHAINFILE,LINKTITLE,IOR);
        IF IOR = 0 THEN
        CHAIN(CHAINFILE)
      ELSE
        WRITELN('  ',LINKTITLE,' not found');
        CLOSE(CHAINFILE,IOR);(* TO FREE UP IOCB *)
        WRITELN('  Type <return>');
     END;

  END;
  UNTIL FALSE;
END.
```

```
(****************************************

    GRAPHICS AND SOUND DEFINITIONS

****************************************)


TYPE
  SCRNTYPE = (SPLIT_SCREEN,FULL_SCREEN);
  CLEAR_TYPE = (CLEAR_SCREEN,DO_NOT_CLEAR_SCREEN);

VAR
  SCRNFILE : EXTERNAL TEXT; (* GRAPHICS FILE *)
  GRRESULT : EXTERNAL INTEGER;
        (* RESULT OF VARIOUS GRAPHICS OPERATIONS:
                INITGRAPHICS    GRRESULT = 0 OK, 255 = ERROR
                GRAPHICS        GRRESULT = 0 OK, 255 = ERROR
                PLOT            GRRESULT = RESULT FROM XIO CALL
                LOCATE          GRRESULT = RESULT FROM XIO CALL
                FILL            GRRESULT = RESULT FROM XIO CALL
                DRAWTO          GRRESULT = RESULT FROM XIO CALL
        *)

EXTERNAL PROCEDURE INITGRAPHICS(MAX_MODE:INTEGER);
        (********************************
        PURP:   INITIALIZE THE GRAPHICS STUFF
        ********************************)

EXTERNAL PROCEDURE GRAPHICS(MODE:INTEGER; SCREEN:SCRN_TYPE; CLEAR:CLEAR_TYPE);
        (**********************************
        PURP:   SET GRAPHICS MODE
        **********************************)

EXTERNAL PROCEDURE TEXTMODE;
        (******************************
        PURP:   RETURN TO STANDARD TEXT MODE
        ******************************)

EXTERNAL PROCEDURE SETCOLOR(REGISTER,HUE,LUMINANCE:INTEGER);
        (**********************************
        PURP:   SET THE HUE AND LIMINANCE OF THE
                SPECIFIED REGISTER
                REGISTER = 0..8
                HUE = 0..15
                LUMINANCE = 0..15 (EVEN ONLY)
        *********************************)


EXTERNAL PROCEDURE COLOR(COLOR_VALUE:INTEGER);
        (**********************************
        PURP:   SET THE CURRENT COLOR
        *********************************)

EXTERNAL PROCEDURE PLOT(X,Y:INTEGER);
        (*********************************
        PURP:   PLOT A POINT AT X,Y OF THE CURRENT COLOR
        *********************************)

EXTERNAL FUNCTION  LOCATE(X,Y:INTEGER):INTEGER;
        (*********************************
        PURP:   RETURN THE CURRENT PIXEL VALUE AT X,Y
```

```
EXTERNAL PROCEDURE POSITION(X,Y:INTEGER);
        (************************************
        PURP:   POSTION THE CURSOR TO X,Y
        ************************************)

EXTERNAL PROCEDURE DRAWTO(X,Y:INTEGER);
        (***********************************
        PURP:   DRAW A LINE IN THE CURRENT COLOR
                TO X,Y
        ************************************)

EXTERNAL PROCEDURE FILL(X,Y:INTEGER);
        (***********************************
        PURP:   DRAW A LINE IN THE CURRENT COLOR
                TO X,Y
        ************************************)

EXTERNAL PROCEDURE SOUND(VOICE,PITCH,DISTORTION,VOLUME:INTEGER);
        (******************************
        PURP:   TURN ON THE SOUND
        ******************************)

EXTERNAL PROCEDURE SOUNDOFF;
        (**************************
        PURP:   TURN OFF THE SOUND
        **************************)

EXTERNAL FUNCTION PADDLE(PDLNUM:INTEGER):INTEGER;
        (**************************
        PURP:   RETURN THE CURRENT PADDLE VALUE
        *************************)

EXTERNAL FUNCTION  PTRIG(PDLNUM:INTEGER):INTEGER;
        (****************************
        PURP:   RETURN CURRENT STATE OF A PADDLE TRIGGER
        ****************************)

EXTERNAL FUNCTION  STICK(STKNUM:INTEGER):INTEGER;
        (**************************
        PURP:   RETURN THE CURRENT STICK VALUE
        *************************)

EXTERNAL FUNCTION  STRIG(STKNUM:INTEGER):INTEGER;
        (****************************
        PURP:   RETURN CURRENT STATE OF A STICK TRIGGER
        ****************************)
```

(*TEXT*)

```
PROGRAM COPIER;
VAR   IOR:INTEGER;
      BUFFER:STRING[255];
      RESPONSE,INFILE,OUTFILE:STRING;
      CHAINFIL:FILE;
      FIN,FOUT:TEXT;
(*$ID:DSKPROCS*)
(*$ID:ISOPROCS*)
BEGIN
RESPONSE:='Y';
REPEAT
        BEGIN
        WRITE('> INPUT FILENAME: ');
        READLN(INFILE);
        WRITE('OUTPUT FILENAME: ');
        READLN(OUTFILE);
        OPEN(FIN,INFILE,IOR);
        ASSIGN(FOUT,OUTFILE);
        REWRITE(FOUT);
        WHILE NOT EOF(FIN) DO
                BEGIN
                READLN(FIN,BUFFER);
                WRITELN(FOUT,BUFFER);
                END;
        CLOSE(FIN,IOR);
        CLOSE(FOUT,IOR);
        WRITE('DO YOU WISH TO DO ANOTHER COPY? (Y/N) ');
        READLN(RESPONSE)
        END
UNTIL RESPONSE[1]='N';
OPEN(CHAINFIL,'D:MON',IOR);
IF IOR<>0 THEN
        BEGIN
        WRITELN('UNABLE TO OPEN D:MON');
        EXIT
        END;
CHAIN(CHAINFIL)
END.
```

(*BYTE*)

```
PROGRAM COPIER;
VAR   TRUTH:BOOLEAN;
      TEMP:BYTE;
      IOR:INTEGER;
      RESPONSE,INFILE,OUTFILE:STRING;
      CHAINFIL:FILE;
      FIN,FOUT:FILE OF BYTE;
(*$ID:DSKPROCS*)
(*$ID:ISOPROCS*)
BEGIN
RESPONSE:='Y';
REPEAT
        BEGIN
        WRITE('> INPUT FILENAME: ');
        READLN(INFILE);
        WRITE('OUTPUT FILENAME: ');
        READLN(OUTFILE);
        OPEN(FIN,INFILE,IOR);
        ASSIGN(FOUT,OUTFILE);
        REWRITE(FOUT);
        WHILE NOT EOF(FIN) DO
                BEGIN
                TEMP:=GNB(FIN);
                TRUTH:=WNB(FOUT,TEMP);
                END;
        CLOSE(FIN,IOR);
        CLOSE(FOUT,IOR);
        WRITE('DO YOU WISH TO DO ANOTHER COPY? (Y/N) ');
        READLN(RESPONSE)
        END
UNTIL RESPONSE[1]='N';
OPEN(CHAINFIL,'D:MON',IOR);
IF IOR<>0 THEN
        BEGIN
        WRITELN('UNABLE TO OPEN D:MON');
        EXIT
        END;
CHAIN(CHAINFIL)
END.
```

```
MODULE CHAINS;
VAR CHAINFIL : FILE;
      RESULT : INTEGER;
(*$ID:DSKPROCS*)
PROCEDURE CHAINER;
BEGIN
OPEN(CHAINFIL,'D:MON',RESULT);
IF RESULT<>0 THEN
   BEGIN
   WRITELN('UNABLE TO OPEN D:MON');
   EXIT
   END;
CHAIN(CHAINFIL)
END;

MODEND.



MODULE CHAINANY;
VAR CHAINFIL : FILE;
      RESULT : INTEGER;
(*$ID:DSKPROCS*)
PROCEDURE CHAINER(FILENAME:STRING);
BEGIN
OPEN(CHAINFIL,FILENAME,RESULT);
IF RESULT<>0 THEN
   BEGIN
   WRITELN('UNABLE TO OPEN ',FILENAME);
   WRITE('PRESS RETURN TO EXIT');
   READLN;
   EXIT
   END;
CHAIN(CHAINFIL)
END;

MODEND.
```

NUMBASE

```
PROGRAM ~~T[illegible]~~:
VAR RESPONSE:STRING:
    DECIMAL.WIDTH.NUMBER:INTEGER:
BEGIN
WHILE RESPONSE<>'NO' DO
BEGIN
WRITE('INPUT NUMBER->');READLN(NUMBER);
WRITE('INPUT WIDTH->');READLN(WIDTH);
WRITE('INPUT DECIMAL BASE ->');READLN(DECIMAL);
WRITELN(NUMBER.'='.NUMBER:WIDTH:DECIMAL);
WRITE('DO ANOTHER?');READLN(RESPONSE)
END
END.
```

```
PROGRAM TSTGRSND:

(******************************
FILE:   TSTGRSND.SRC
PURP:   TEST GRAPHICS AND SOUND
CHANGES:
        2/9/91 (WLS)
                CREATED

        4/15/81 [John Eckstrom]
                change include statement
                to reflect drive #
        4/21/82 <DAVID GETREU>
                ADDED ABILITY TO
                CHAIN BACK TO 'D:MON'
******************************)
VAR     CMD : STRING:
(*$I D:GSPROCS*)

EXTERNAL PROCEDURE CHAINER:

PROCEDURE GETXYCO(VAR X,Y:INTEGER):
BEGIN
  WRITE('ENTER X COORD: '): READLN(X):
  WRITE('ENTER Y COORD: '): READLN(Y)
END:

PROCEDURE GETPDL(VAR PDL:INTEGER):
BEGIN
  WRITE('ENTER PADDLE NUMBER: '):
  READLN(PDL)
END:

PROCEDURE GETSTICK(VAR STICK:INTEGER):
BEGIN
  WRITE('ENTER STICK NUMBER: '):
  READLN(STICK)
END:

PROCEDURE WRTGRRE:
VAR GRRESULT:INTEGER:
BEGIN
  WRITELN('GRRESULT=',GRRESULT)
END:

PROCEDURE CMDGRAP:
VAR
  MODE          : INTEGER:
  STYPE, CTYPE  : STRING:
  SCRNTIPE      : SCRN_TYPE:
  CLRTIPE       : CLEAR_TYPE;

BEGIN
  WRITE('ENTER MODE: '): READLN(MODE):
  WRITE('ENTER FULL/SPLIT SCREEN [F/S]: '): READLN(STYPE);
  IF STYPE = 'F'
  THEN
       SCRNTIPE := FULL_SCREEN
  ELSE
       SCRNTIPE := SPLIT_SCREEN;
  WRITE('ENTER CLEAR/NO CLR SCRN [C/N]: '): READLN(CTYPE):
      CLRTIPE := CLEAR_SCREEN
  ELSE
      CLRTIPE := DO_NOT_CLEAR_SCREEN;
  GRAPHICS(MODE.SCRNTIPE.CLRTIPE);
  WRTGRRE
END;

PROCEDURE CMDSETCO;
VAR
  REG.HUE,LUM:INTEGER;
BEGIN
  WRITE('ENTER REGISTER: '); READLN(REG);
  WRITE('ENTER HUE: '); READLN(HUE);
  WRITE('ENTER LUMINANCE: '); READLN(LUM);
  SETCOLOR(REG.HUE,LUM)
END;

PROCEDURE CMDPOS;
VAR
  X.Y:INTEGER;
BEGIN
  WRITELN('POSITION');
  GETXYCO(X,Y);
  POSITION(X.Y)
END;

PROCEDURE CMDPLOT;
VAR
  X.Y:INTEGER;
BEGIN
  WRITELN('PLOT');
  GETXYCO(X,Y);
  PLOT(X.Y);
  WRTGRRE
END;

PROCEDURE CMDCOLR;
VAR
  COLORVALUE:INTEGER;
BEGIN
  WRITE('ENTER COLOR VALUE: '); READLN(COLORVALUE);
  COLOR(COLORVALUE)
END;

PROCEDURE CMDLOCA;
VAR
  X.Y:INTEGER;
BEGIN
  WRITELN('LOCATE');
  GETXYCO(X,Y);
  WRITELN('LOCATE(',X,',',Y,')=',LOCATE(X,Y));
  WRTGRRE
END;

PROCEDURE CMDDRAW;
VAR
  X.Y:INTEGER;
BEGIN
  WRITELN('DRAWTO');
  GETXYCO(X,Y);
  DRAWTO(X.Y);
  WRTGRRE
END;
```

```
VAR
  X,Y:INTEGER;
BEGIN
  WRITELN('FILL');
  GETXYCO(X,Y);
  FILL(X,Y);
  WRTGRRE
E[illegible]

PROCEDURE CMDSOUND;
VAR
  VOICE,PITCH,DISTORTION,VOLUME:INTEGER;
BEGIN
  WRITE('ENTER VOICE: '); READLN(VOICE);
  WRITE('ENTER PITCH: '); READLN(PITCH);
  WRITE('ENTER DISTORTION: '); READLN(DISTORTION);
  WRITE('ENTER VOLUME: '); READLN(VOLUME);
  SOUND(VOICE,PITCH,DISTORTION,VOLUME)
END;

PROCEDURE CMDPAD;
VAR
  PDLNUM:INTEGER;
BEGIN
  GETPDL(PDLNUM);
  WRITELN('PADDLE(',PDLNUM,')=',PADDLE(PDLNUM))
END;

PROCEDURE CMDPTRIG;
VAR
  PDLNUM:INTEGER;
BEGIN
[illegible]TPDL(PDLNUM);
  [illegible]RITELN('PTRIG(',PDLNUM,')=',PTRIG(PDLNUM))
END;

PROCEDURE CMDSTICK;
VAR
  PDLNUM:INTEGER;
BEGIN
  GETSTICK(PDLNUM);
  WRITELN('STICK(',PDLNUM,')=',STICK(PDLNUM))
END;

PROCEDURE CMDSTRIG;
VAR
  PDLNUM:INTEGER;
BEGIN
  GETSTICK(PDLNUM);
  WRITELN('STRIG(',PDLNUM,')=',STRIG(PDLNUM))
END;



BEGIN    (*  MAIN PROGRAM   *)
INITGRAPHICS(8);
WRITE('INITGRAPHICS ');
W[illegible]GRRE;
R[illegible]EAT
   WRITE('ENTER COMMAND: ');
   READLN(CMD);
   IF CMD = 'GRAPHICS' THEN CMDGRAP;
   IF CMD = 'TEXTMODE' THEN TEXTMODE;
   IF CMD = 'SETCOLOR' THEN CMDSETCO;
   IF CMD = 'POSITION' THEN CMDPOS;
   IF CMD = 'LOCATE'    THEN CMDLOCA;
   IF CMD = 'FILL'      THEN CMDFILL;
   IF CMD = 'DRAW'      THEN CMDDRAW;
   IF CMD = 'SOUND'     THEN CMDSOUND;
   IF CMD = 'SOUNDOFF'  THEN SOUNDOFF;
   IF CMD = 'PADDLE'    THEN CMDPAD;
   IF CMD = 'PTRIG'     THEN CMDPTRIG;
   IF CMD = 'STICK'     THEN CMDSTICK;
   IF CMD = 'STRIG'     THEN CMDSTRIG
UNTIL CMD = 'EXIT';
CHAINER
END.
```